

PROGRAMAÇÃO WEB

Curso Superior de Tecnologia em Análise e
Desenvolvimento de Sistemas

Prof. Edécio Fernando Iepsen

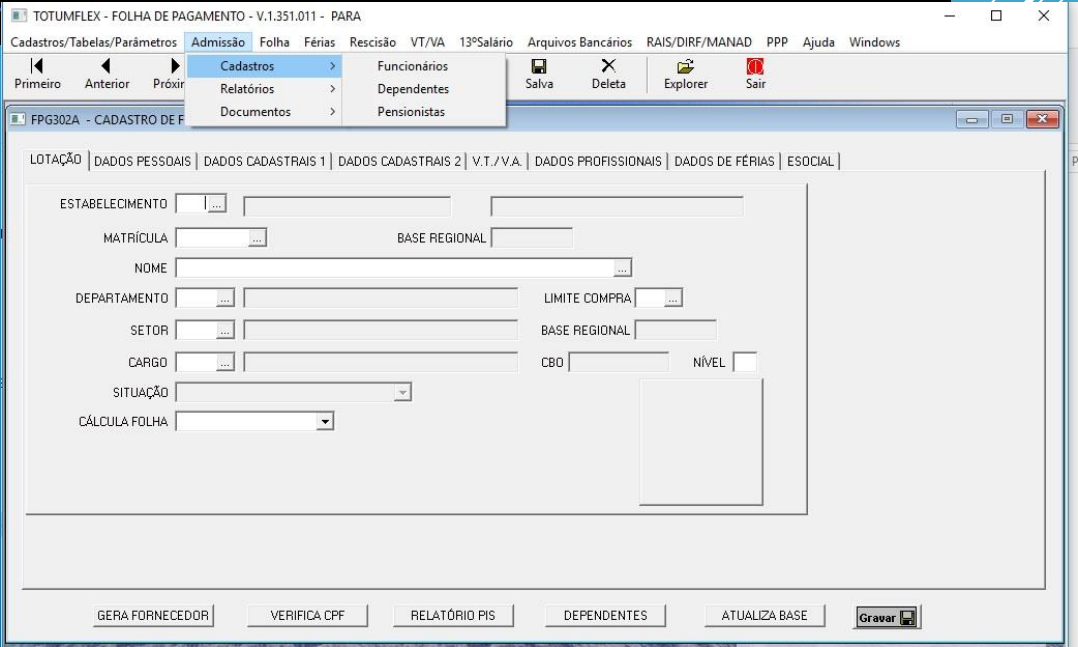
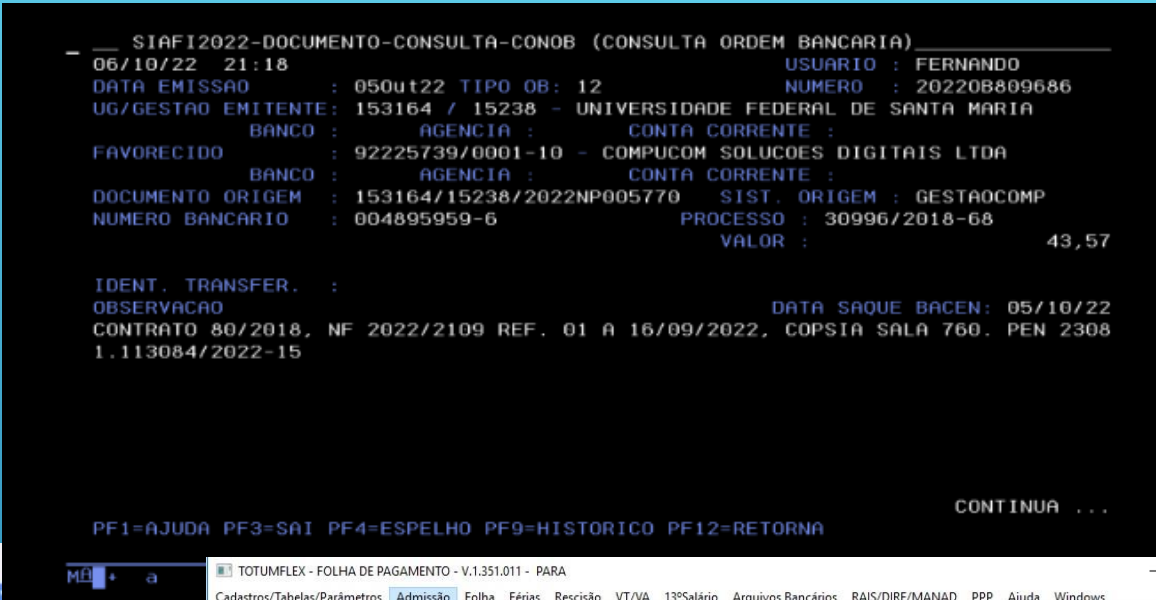
- Lógica de Programação
- Desenvolvimento de Interfaces Web

C: > logica232_manha > repeticoes3 > JS revenda.js > ...

```
1  const prompt = require("prompt-sync")()
2
3  let contador = 0
4  let continua
5  do {
6    const modelo = prompt("Modelo: ")
7    const preco = Number(prompt("Preço R$: "))
8    if (preco > 30000) {
9      contador = contador + 1      // ou contador++
10   }
11
12   // const prDesconto = preco - (preco * 0.10)
13   const prDesconto = preco * 0.90
14   console.log(`${modelo} com Desconto R$: ${prDesconto.toFixed(2)}`)
15   continua = prompt("Continua (S/N): ").toUpperCase()
16 } while (continua == "S")
17
18 if (contador == 0) {
19   console.log("\nSó carros baratinhos... :)")
20 } else {
21   console.log(`\n${contador} carros com preço superior a R$ 30000`)
22 }
```

```
1  <!DOCTYPE html>
2  <html lang="pt-br">
3  <head>
4    <meta charset="UTF-8">
5    <meta http-equiv="X-UA-Compatible" content="IE=edge">
6    <meta name="viewport" content="width=device-width, initial-scale=1.0">
7    <link rel="stylesheet" href="estilos.css">
8    <link rel="shortcut icon" href="logo.webp" type="image/x-icon">
9    <title>Clube de Nataç o</title>
10 </head>
11 <body>
12   <div class="container">
13     <header>
14       
15       <nav>
16         <ul>
17           <li><a href="#link-avisos">Avisos</a></li>
18           <li><a href="#">Hor rios</a></li>
19           <li><a href="#">Associe-se</a></li>
20         </ul>
21       </nav>
22     </header>
23     <section class="pos-header">
24       
25       <h1>Clube Avenida - seu Clube de Nata o</h1>
26       <p>H  25 anos incentivando o esporte aqu tico na cidade de Pelotas</p>
27       <p class="esporte">Esporte   vida! Associe-se e fa a parte de nosso clube.</p>
28     </section>
29     <span id="link-avisos"></span>
30     <section class="avisos">
31       <h2>Avisos aos associados do Clube Avenida</h2>
32       <ol>
33         <li>Lorem ipsum, dolor sit amet consectetur adipisicing elit. Aut dicta praesentium d
34         <li>Lorem ipsum dolor sit <span class="esporte">amet consectetur, adipisicing elit</s
```

Aplicação dos Conceitos de Lógica de Programação




Programação Web: Exemplos

Tênis Olympikus Reverb Unissex

olympikus.com.br/tenis-olympikus-reverb-unissex-43691174-3-030/p




Atualizar

Parcele suas compras em até 10x sem juros





MASCULINO FEMININO INFANTIL LANÇAMENTOS OFERTAS BPC CORRE-JUNTO

O QUE VOCÊ ESTÁ PROCURANDO?




Página Inicial / Masculino / Tênis / Casual

32% OFF



FEITO POR BRASILEIROS

TÊNIS OLYMPIKUS REVERB UNISSEX



★★★★★ (23) Ref.:43691174_3-030-34


De R\$ 249,99 por

R\$ 169,99

ou até 10x de R\$ 16,99 [Veja Parcelas](#)


Economize R\$ 80,00



7 CORES DISPONÍVEIS




TAMANHO:

34 35 37 38 39 40 41 42 43 44

 1659 pessoas viram esse produto hoje!

 Provador virtual  Tabela de medidas

QUANTIDADE: 01



Programação Web: Exemplos



The screenshot shows a web browser window with the URL `qbemqfaz.com.br/nutricionistas/imc-calculadora#calculatorIMC`. The website has a green header with navigation links: **Nutricionistas**, **Colaboradores**, **Entrar**, and **Criar conta**. Below this is a light gray navigation bar with links: **Saúde**, **Bem-Estar**, **Alimentação**, **Vida Equilibrada**, **Receitas**, and **Quem Somos?**. The main content area is titled **Calcular IMC** and includes a large image of a foot on a pink scale. The text explains that to calculate BMI, users need to enter height and weight. There are two input fields: **Altura em cm** (with placeholder 'Inserir a altura em cm') and **Peso em kg** (with placeholder 'Inserir o peso em kg'). An example is provided: 'Exemplo: 1,80m de altura = 180cm'. A note states: 'A altura e o peso do paciente devem ser arredondados para melhor funcionamento da ferramenta.' A blue button labeled **Calcular IMC** is at the bottom of the form.

que bem que faz

Calcular IMC

Para medir o IMC com a nossa calculadora, é só digitar os dados nos campos abaixo e clicar em "Calcular". Simples assim!

Altura em cm
Inserir a altura em cm

Peso em kg
Inserir o peso em kg

Exemplo: 1,80m de altura = 180cm

A altura e o peso do paciente devem ser arredondados para melhor funcionamento da ferramenta.

Calcular IMC

O resultado do cálculo do IMC costuma ser classificado de acordo com uma tabela (a famosa tabela de IMC). Ela funciona assim:



Programação Web: Exemplos

Cinema Cineflix Shopping Pelot

cineflix.com.br/programacao/shopping-pelotas/?gclid=Cj0KCQjw3JanBhCPARIsAJpXTx6aUbfokD1xjHAYIG78hDco3UOK16YYps_dqMLy99K6vyEA_QybboaAmCrEALw_wcB

cineflix CINEMAS

Divirta-se!

PROGRAMAÇÃO VENDA ON-LINE EM CARTAZ

Shopping Pelotas

dom 11 seg 12 ter 13 qua 14 qui 15 sex 16 sáb 17

Divertida Mente 2 DUB 16:50

Divertida Mente 2

A sede dos sentimentos (mente) de Riley está passando por uma demolição repentina para dar lugar a novas emoções!

ANIMAÇÃO : 01h36

Diretor: Pete Docter, Kelsey Mann

Estrelando :

Título Original: Inside Out 2, **Ano**: 2024, **Class**: Livre, **Idioma**: Inglês, **Sala**: 5

Divertida Mente 2 | Teaser Trailer Oficial

TEASER DUBLADO

Meu Malvado Favorito 4 DUB 14:40 17:00

Armadilha ★ DUB 14:30 19:10

Frameworks Web JavaScript



React JS

NEXT.js

Express **JS**



Vue.js



ANGULARJS
by Google



nest



Quem somosFormaçãoPagamentoPara empresasDúvidasBlogConteúdos gratuitos

QUERO SER PROGRAMADOR

O QUE SÃO FRAMEWORKS E POR QUE SÃO IMPORTANTES PARA OS DEVS



Por Daniel Kriger | 27/05/22 | 11 min de leitura

Início » Programação

Construir uma aplicação do zero é um trabalho que todo [programador](#) ou programadora já desempenhou.

Quando sua função, porém, é construir diversas vezes o mesmo tipo de aplicação, reescrever a mesma estrutura de código de novo e de novo pode ser meio cansativo, certo?

Ainda que o conhecimento e a experiência permitam que o trabalho seja mais ágil, o tempo despendido para refazer o mesmo trabalho muitas vezes poderia ser melhor aproveitado em outras frentes.

Como uma forma de auxiliar os programadores(as), foram criados os frameworks.

React JS



O React é uma biblioteca front-end JavaScript, criada em 2011 pelo Facebook, focada no desenvolvimento de interfaces de usuário para páginas web.

A criação das views em React utiliza como base os componentes, cujo intuito principal é otimizar a atualização e sincronização das páginas, além de melhorar a manutenção do código.

É mantido pelo Facebook, Instagram, outras empresas e uma comunidade de desenvolvedores individuais.

Em 2015, o Facebook anunciou o módulo React Native, que em conjunto com o React, possibilita o desenvolvimento de aplicativos para Android e iOS.

Hospedagem
de Site Premium

O Que é React e Como Funciona?



O **React** é a biblioteca mais popular do **JavaScript** e é usada para construir uma interface de usuário (IU). Ela oferece uma resposta excelente para o usuário adicionar comandos usando um novo método de renderizar sites.

React Components

Header

Moving
Image

Form

~~~~~

~~~~~

~~~~~

~~~~~ ☐ ~~~~~ ☐

~~~~~ ☐ ~~~~~ ☐

~~~~~

Text

Image

Card

Footer



- In here we are using same 'Text', 'Image' and 'Card' components.
- We are reusing these components. Reducing code lines.



React

A JavaScript library for building user interfaces

Get Started

Take the Tutorial >

Declarativo

O React facilita a criação de UIs interativas. Projete exibições simples para cada estado em seu aplicativo e o React atualizará e renderizará com eficiência apenas os componentes certos quando seus dados mudarem.

Visualizações declarativas tornam seu código mais previsível e fácil de depurar.

Baseado em componentes

Crie componentes encapsulados que gerenciam seu próprio estado e, em seguida, componha-os para criar interfaces de usuário complexas.

Como a lógica do componente é escrita em JavaScript em vez de modelos, você pode facilmente passar dados avançados por meio de seu aplicativo e manter o estado fora do DOM.

Aprenda uma vez, escreva em qualquer lugar

Não fazemos suposições sobre o restante de sua pilha de tecnologia, para que você possa desenvolver novos recursos no React sem reescrever o código existente.

O React também pode renderizar no servidor usando Node e alimentar aplicativos móveis usando [React Native](#).



React Docs

BETA



Q Procurar



K

Aprender

Referência

Iniciar

Instalação



Começo rápido



Aprenda Reagir

Descrevendo a IU



Adicionando interatividade



Gerenciando Estado



Escotilhas de fuga



Sobre Reagir

Comunidade



Esta página é útil?



React Docs

BETA

Começo rápido

Aprenda a pensar em React com explicações passo a passo e exemplos interativos.

[consulte Mais informação >](#)

Referência da API

Procure a API de React Hooks e veja sua forma com assinaturas codificadas por cores.

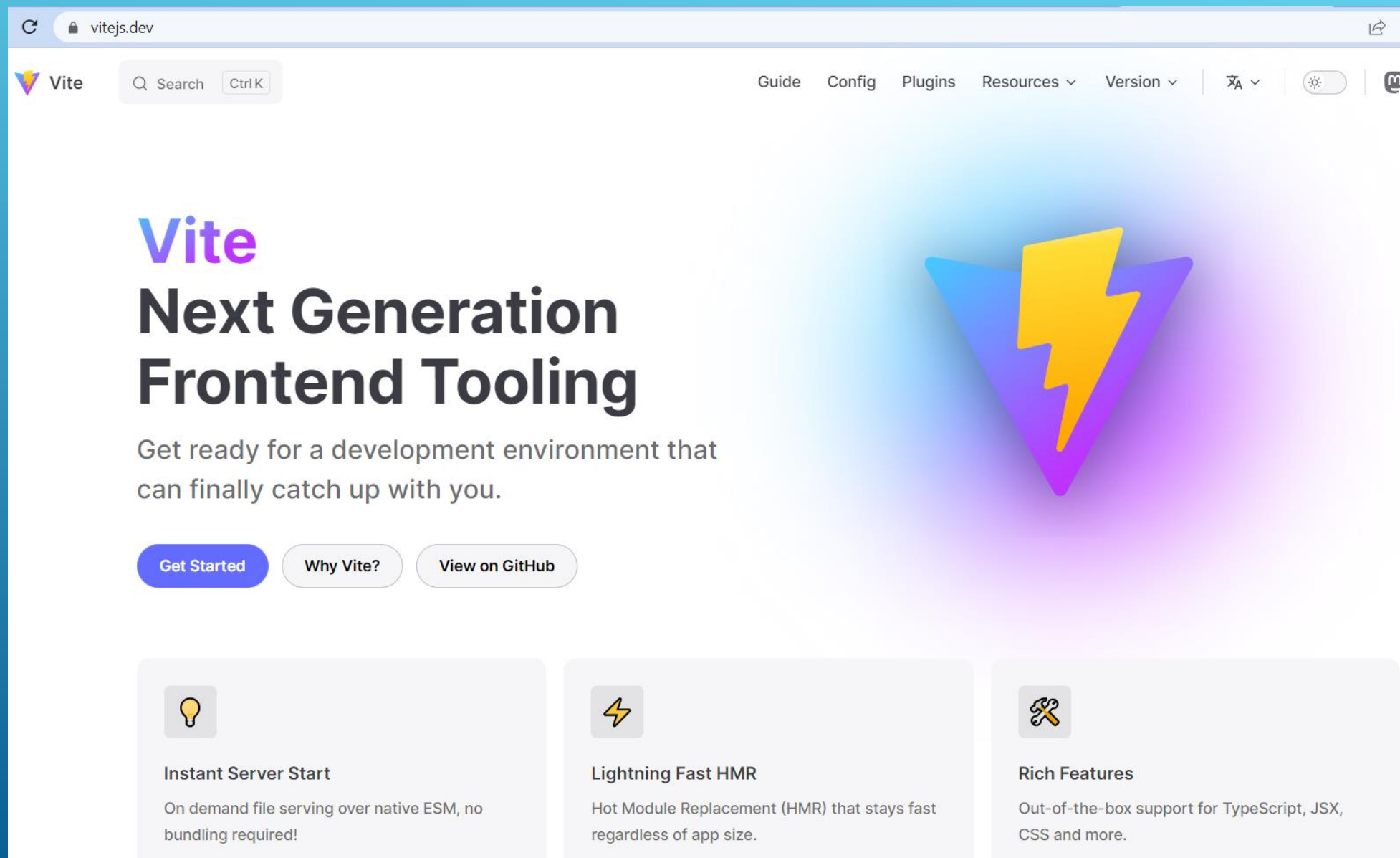
[consulte Mais informação >](#)

O que é este site?

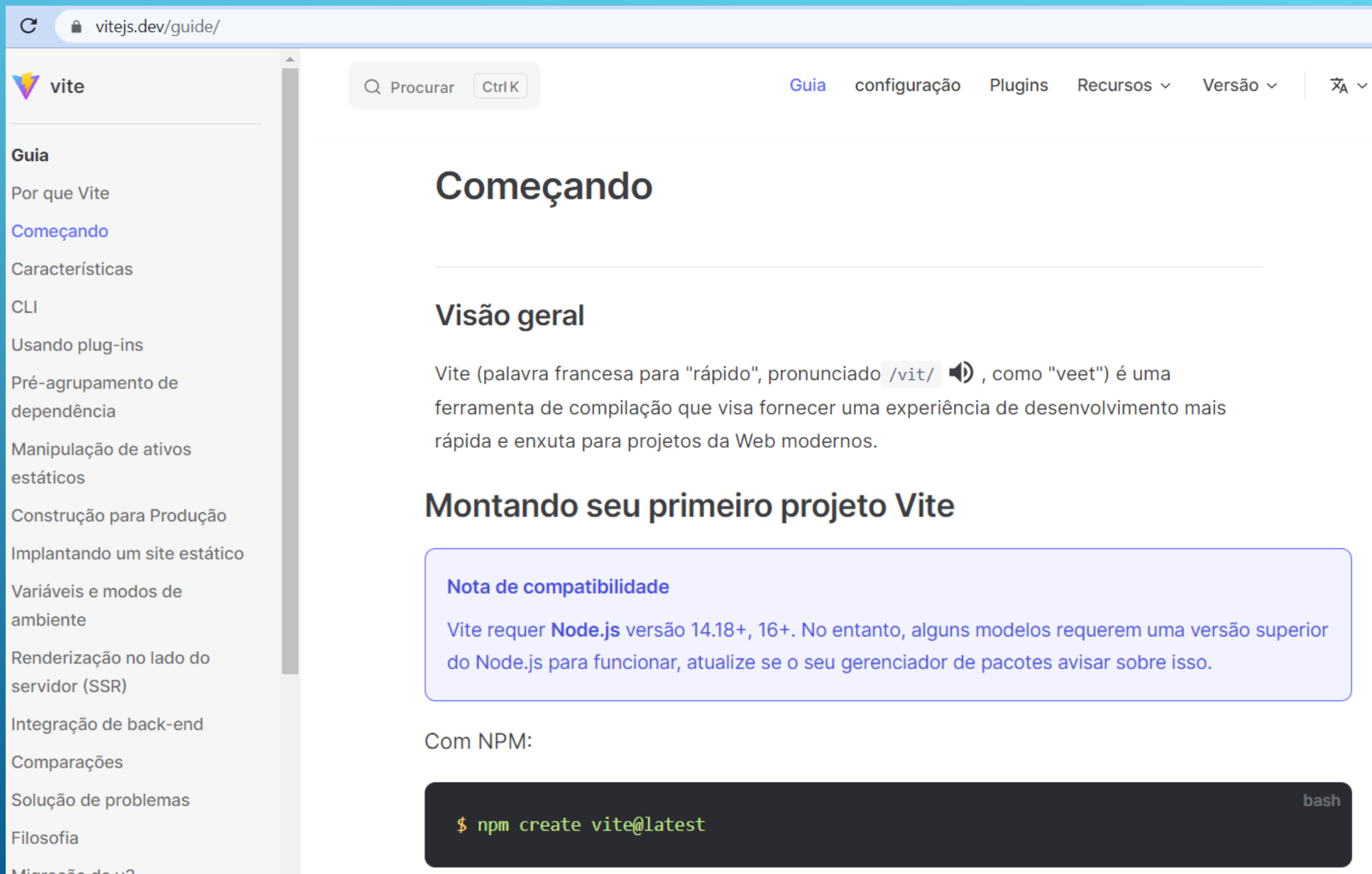
Estamos reescrevendo a documentação do React com algumas diferenças:

- Todas as explicações são escritas usando Hooks em vez de classes.
- Adicionamos exemplos interativos e diagramas visuais.
- Os guias incluem desafios (com soluções!) para verificar sua compreensão.

Criação de novos projetos



Criação de novos projetos



The screenshot shows the Vite.js guide website. The left sidebar contains a navigation menu with the following items: Guia, Por que Vite, Começando (highlighted in blue), Características, CLI, Usando plug-ins, Pré-agrupamento de dependência, Manipulação de ativos estáticos, Construção para Produção, Implantando um site estático, Variáveis e modos de ambiente, Renderização no lado do servidor (SSR), Integração de back-end, Comparações, Solução de problemas, Filosofia, and Migração da v2. The main content area is titled 'Começando' and includes a 'Visão geral' section. The 'Visão geral' section states that Vite is a French word for 'fast', pronounced /vit/ (like 'veet'), and is a compilation tool designed to provide a faster and more concise development experience for modern web projects. Below this is a section titled 'Montando seu primeiro projeto Vite'. A light blue box contains a 'Nota de compatibilidade' (Compatibility Note) stating that Vite requires Node.js version 14.18+, 16+. It also mentions that some models require a higher version of Node.js to function and advises users to check their package manager for updates. At the bottom, under the heading 'Com NPM:', there is a terminal snippet showing the command to create a new Vite project: `$ npm create vite@latest`. The terminal window is dark with a 'bash' prompt on the right.

vitejs.dev/guide/

vite

Procurar CtrlK

Guia configuração Plugins Recursos Versão

Começando

Visão geral

Vite (palavra francesa para "rápido", pronunciado /vit/ , como "veet") é uma ferramenta de compilação que visa fornecer uma experiência de desenvolvimento mais rápida e enxuta para projetos da Web modernos.

Montando seu primeiro projeto Vite

Nota de compatibilidade

Vite requer **Node.js** versão 14.18+, 16+. No entanto, alguns modelos requerem uma versão superior do Node.js para funcionar, atualize se o seu gerenciador de pacotes avisar sobre isso.

Com NPM:

```
$ npm create vite@latest
```


bash

Criação de novos projetos

```
C:\aulas24\react>npm create vite@latest sinaleira
Need to install the following packages:
create-vite@5.2.1
Ok to proceed? (y) y
? Select a framework: » - Use arrow-keys. Return to submit.
  Vanilla
  Vue
>  React
  Preact
  Lit
  Svelte
  Solid
  Qwik
  Others
```

npm create vite@latest nomeproj

Criação de novos projetos

```
C:\aulas24\react>npm create vite@latest sinaleira
Need to install the following packages:
create-vite@5.2.1
Ok to proceed? (y) y
✓ Select a framework: » React
? Select a variant: » - Use arrow-keys. Return to submit.
  TypeScript
  TypeScript + SWC
> JavaScript
  JavaScript + SWC
  Remix 
```

Criação de novos projetos

```
C:\aulas24\react>npm create vite@latest sinaleira
Need to install the following packages:
create-vite@5.2.1
Ok to proceed? (y) y
✓ Select a framework: » React
✓ Select a variant: » JavaScript

Scaffolding project in C:\aulas24\react\sinaleira...

Done. Now run:

cd sinaleira
npm install
npm run dev
```

Criação de novos projetos

```
C:\aulas24\react>cd sinaleira

C:\aulas24\react\sinaleira>npm install

added 273 packages, and audited 274 packages in 38s

98 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\aulas24\react\sinaleira>npm run dev

> sinaleira@0.0.0 dev
> vite

VITE v5.1.4  ready in 537 ms

  Local:   http://localhost:5173/
  Network: use --host to expose
  press h + enter to show help
```

Criação de novos projetos

```
VITE v5.1.4 ready in 537 ms
```

```
? Local: http://localhost:5173/
```

```
? Network: use --host to expose
```

```
? press h + enter to show help
```

h

Shortcuts

press r + enter to restart the server

press u + enter to show server url

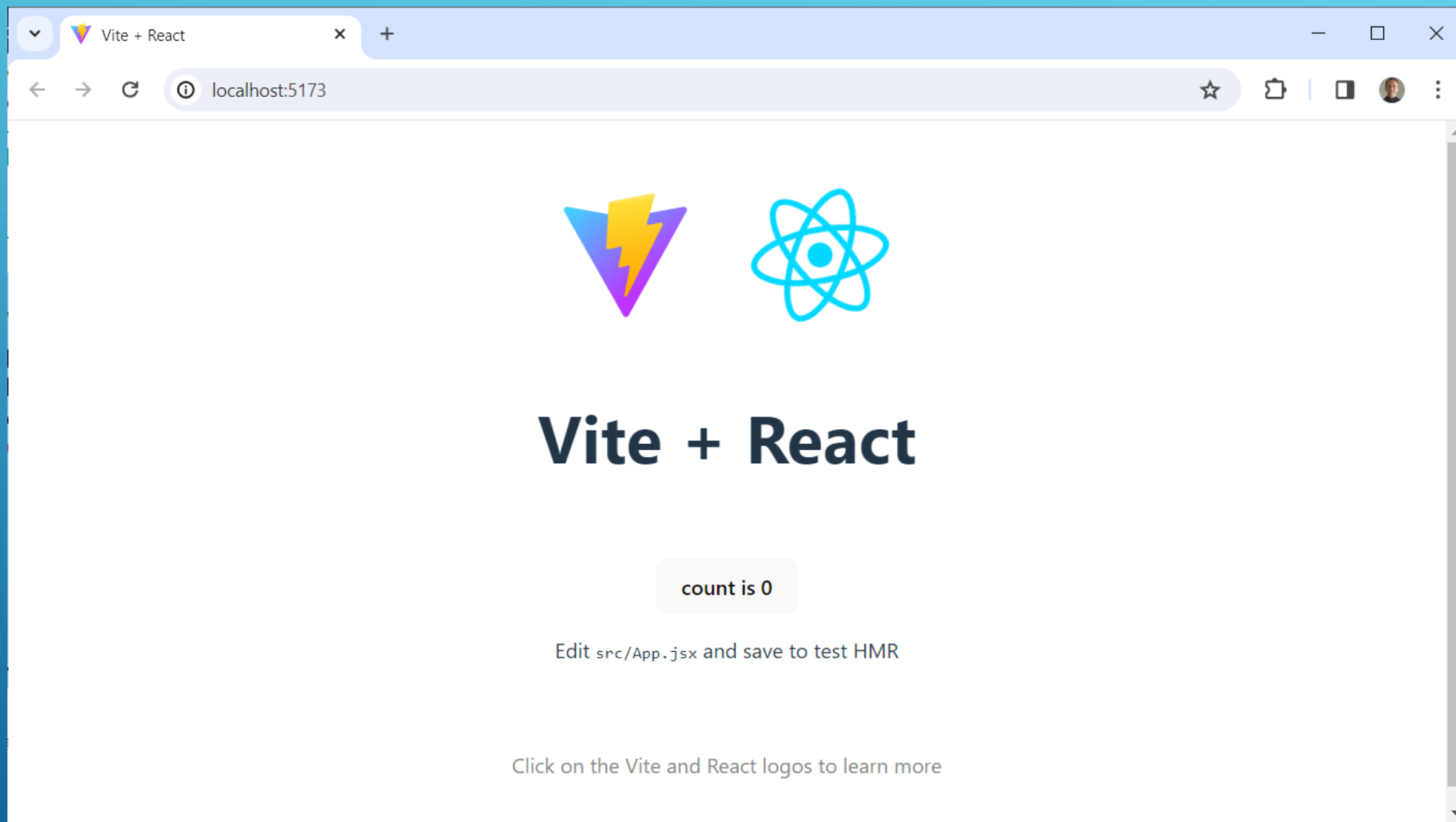
press o + enter to open in browser

press c + enter to clear console

press q + enter to quit

o

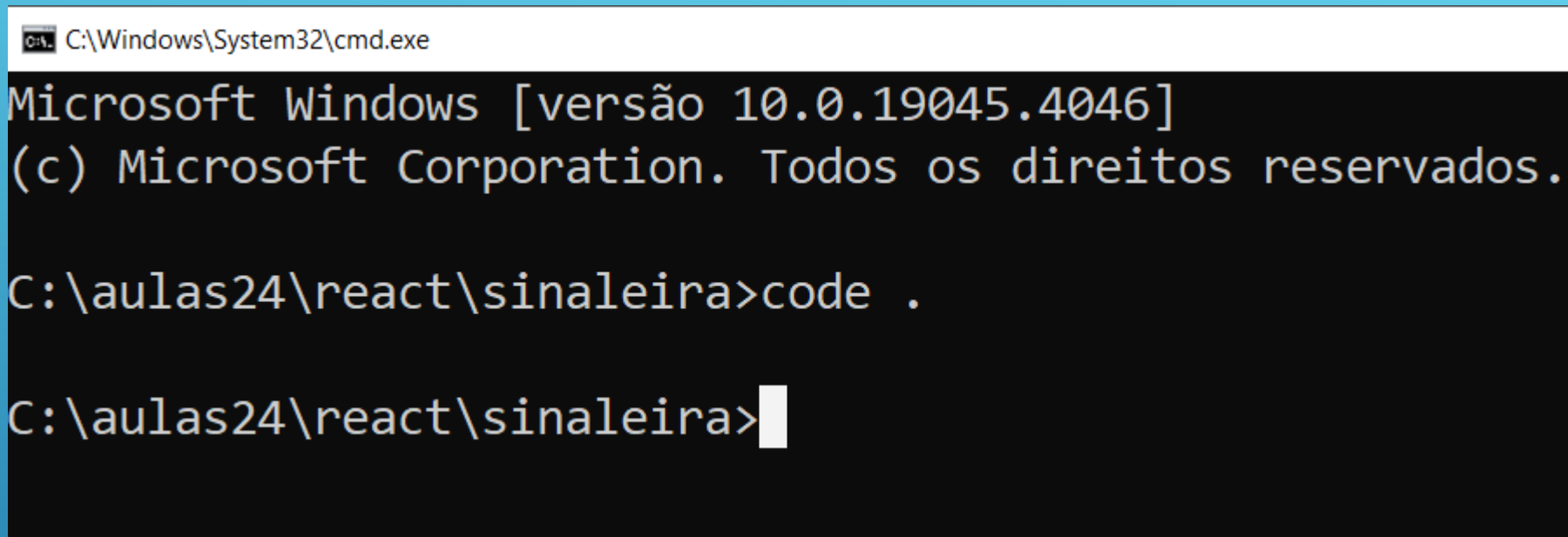
Criação de novos projetos



Observações:

- Deve-se sempre executar todos esses passos para criar cada novo projeto.
- Todos os novos projetos em React já irão conter um conjunto de arquivos e uma estruturação inicial de pastas. Nós modificamos esses arquivos para criar a nossa aplicação.
- Pode parecer “estranho”, pois, tanto em Lógica de Programação, quanto em Desenvolvimento de Interfaces Web, criávamos um projeto a partir de um arquivo vazio. Mas, todos os frameworks são assim.

Criação de novos projetos



```
C:\Windows\System32\cmd.exe
Microsoft Windows [versão 10.0.19045.4046]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\aulas24\react\sinaleira>code .

C:\aulas24\react\sinaleira>
```

- Abra um novo prompt de comandos para entrar no VSCode
- Obs.: O prompt com o comando `npm run dev` precisa permanecer em execução

Pastas e arquivo index.html

Nome	Data de modificação
node_modules	19/01/2024 19:42
public	19/01/2024 20:34
src	19/01/2024 20:36
.eslintrc.cjs	08/01/2024 14:49
.gitignore	08/01/2024 14:49
index.html	
package.json	
package-lock.json	
README.md	
vite.config.js	

<> index.html X

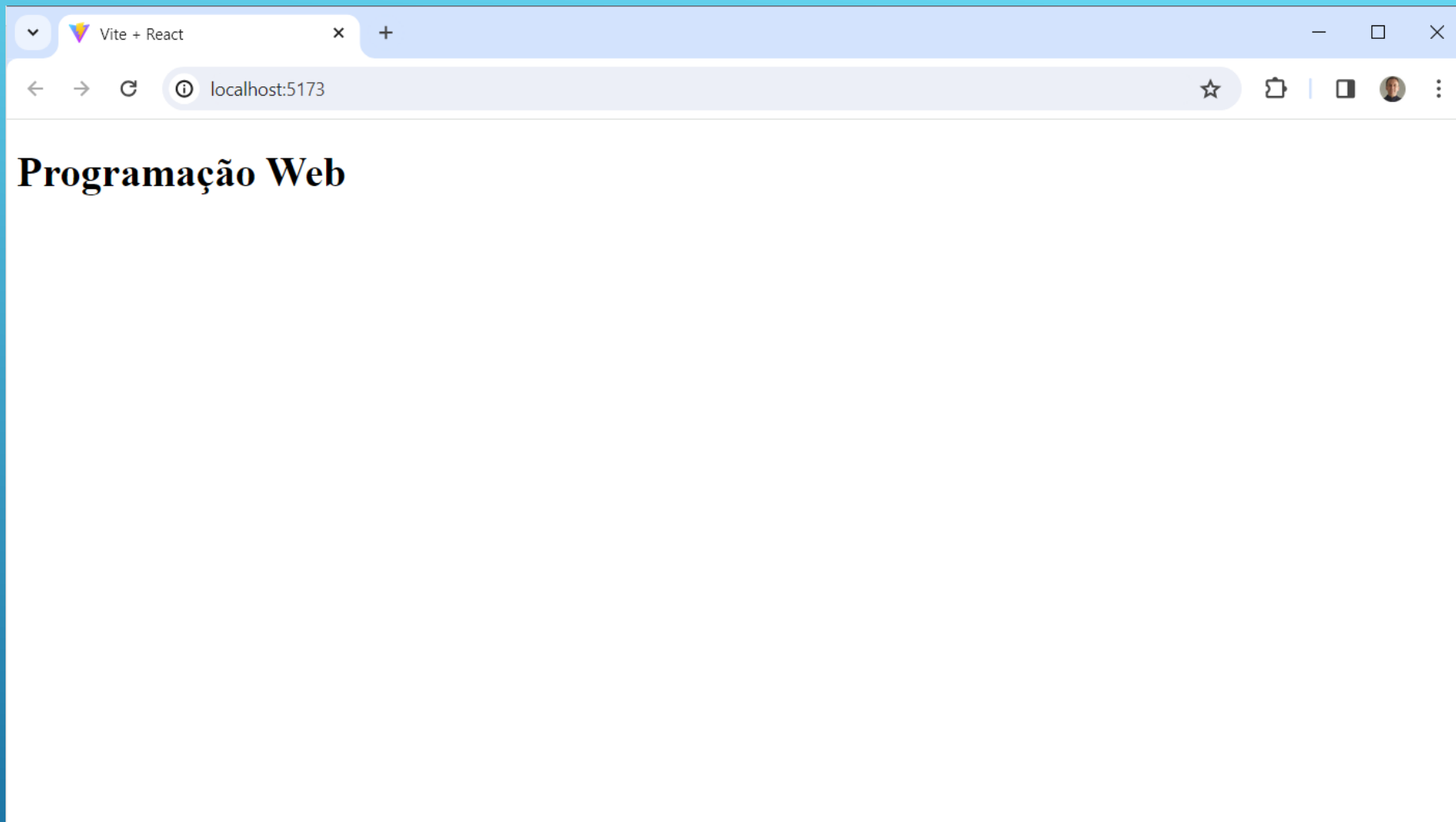
<> index.html > ...

```
1  <!doctype html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <link rel="icon" type="image/svg+xml" href="/vite.svg" />
6      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7      <title>Vite + React</title>
8    </head>
9    <body>
10     <div id="root"></div>
11     <script type="module" src="/src/main.jsx"></script>
12   </body>
13 </html>
14
```


Arquivos iniciais da pasta src

```
main.jsx × App.jsx
src > main.jsx
1  import React from 'react'
2  import ReactDOM from 'react-dom/client'
3  import App from './App.jsx'
4
5  ReactDOM.createRoot(document.getElementById('root')).render(
6    <React.StrictMode>
7    |   <App />
8    </React.StrictMode>,
9  )
```

```
main.jsx App.jsx ×
src > App.jsx > ...
1  function App() {
2    |   return (
3    |     <h1>Programação Web</h1>
4    |   )
5  }
6
7  export default App
8
```

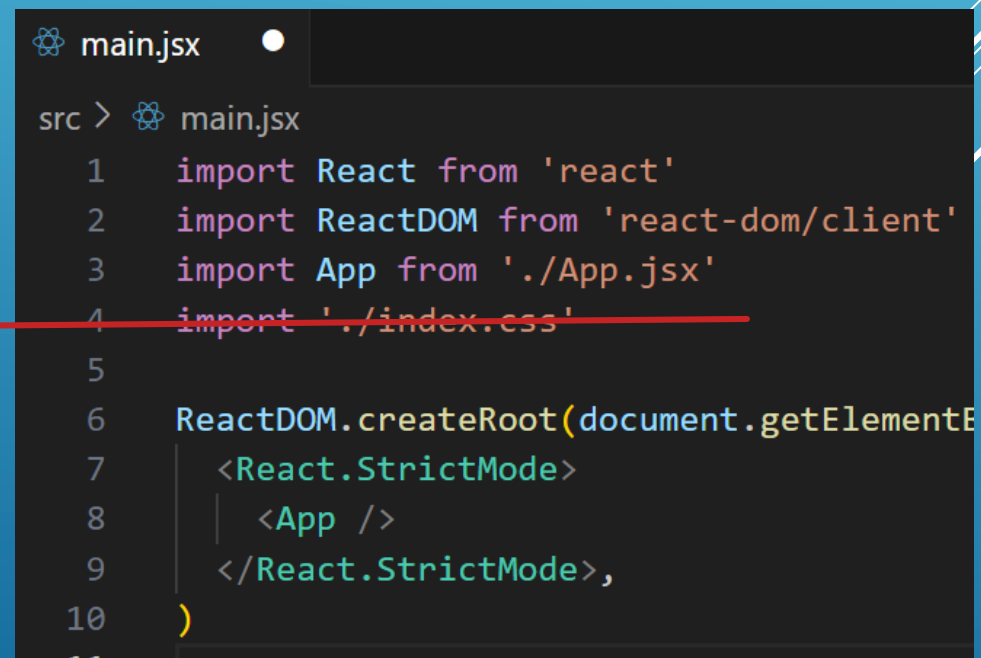


Tudo isso, para só isso...



Importante:

- Inicialmente, só iremos programar no arquivo App.jsx
- Em App.jsx, um elemento (tag HTML) deve envolver todos os demais
- No arquivo index.html pode-se ajustar o idioma, título e ícone
- Os arquivos .css (index.css e App.css) podem ser removidos
- No arquivo main.jsx deve-se remover a linha 4 (`import './index.css'`)
- Imagens podem ser inseridas na pasta public e são referenciadas no código como `“./nomearq.ext”`



```
main.jsx
src > main.jsx
1  import React from 'react'
2  import ReactDOM from 'react-dom/client'
3  import App from './App.jsx'
4  import './index.css'
5
6  ReactDOM.createRoot(document.getElementById('root')).render(
7    <React.StrictMode>
8      <App />
9    </React.StrictMode>,
10 )
```

Ajustes no index.html

<> index.html X

<> index.html > ...

```
1  <!doctype html>
2  <html lang="pt-br">
3    <head>
4      <meta charset="UTF-8" />
5      <link rel="icon" type="image/svg+xml" href="./logo.png" />
6      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7      <title>Restaurante Avenida</title>
8    </head>
9    <body>
10     <div id="root"></div>
11     <script type="module" src="/src/main.jsx"></script>
12   </body>
13 </html>
14 |
```

Escrevendo marcação com JSX

```
function AboutPage() {  
  return (  
    <>  
      <h1>About</h1>  
      <p>Hello there.<br />How do you do?</p>  
    </>  
  );  
}
```

A sintaxe de marcação que você viu acima é chamada *JSX*. É opcional, mas a maioria dos projetos React usa JSX para sua conveniência. Todas as [ferramentas que recomendamos para desenvolvimento local](#) oferecem suporte a JSX pronto para uso.

JSX é mais rígido que HTML. Você tem que fechar tags como `
`. Seu componente também não pode retornar várias tags JSX. Você precisa envolvê-los em um pai compartilhado, como um wrapper `<div>...</div>` vazio: `<>...</>`

Adicionando estilos

No React, você especifica uma classe CSS com `className`. Funciona da mesma forma que o `class` atributo HTML:

```
<img className="avatar" />
```

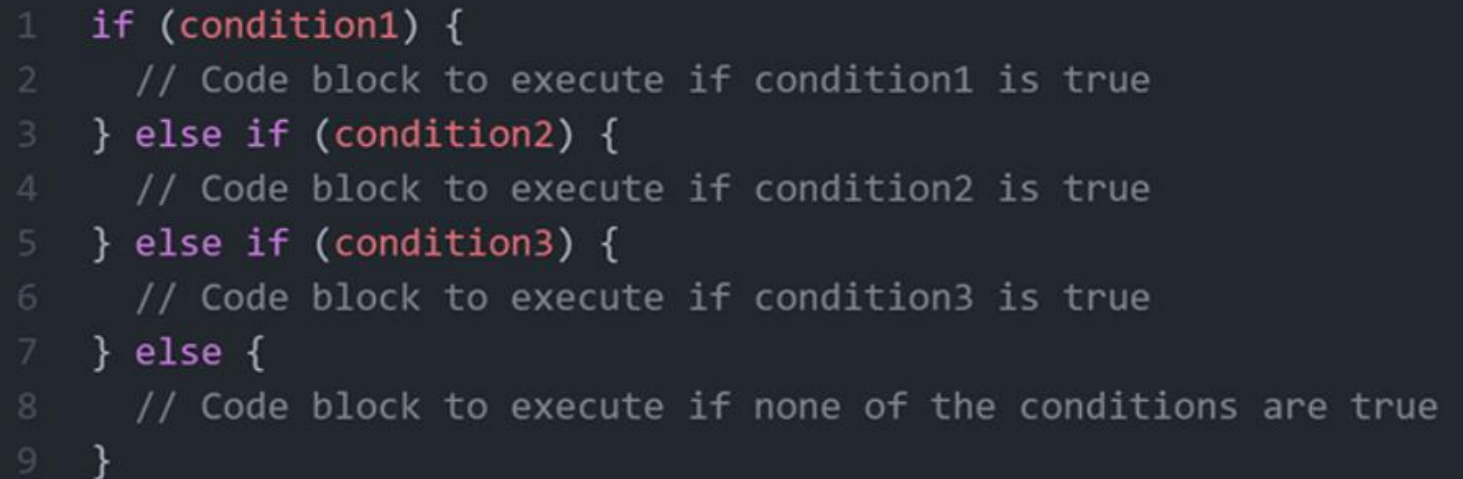
Então você escreve as regras CSS para ele em um arquivo CSS separado:

```
/* In your CSS */  
.avatar {  
  border-radius: 50%;  
}
```

Revisando... Variáveis

- As variáveis servem para armazenar os dados manipulados pelo programa.
- Elas ocupam um espaço em memória e permitem saber, por exemplo, o que o usuário digitou.
- As variáveis precisam ser identificadas por um nome.

Condições



```
1  if (condition1) {  
2      // Code block to execute if condition1 is true  
3  } else if (condition2) {  
4      // Code block to execute if condition2 is true  
5  } else if (condition3) {  
6      // Code block to execute if condition3 is true  
7  } else {  
8      // Code block to execute if none of the conditions are true  
9  }
```

Exibindo dados

JSX permite colocar marcação em JavaScript. As chaves permitem que você “volte” para o JavaScript para que você possa incorporar alguma variável do seu código e exibi-la ao usuário. Por exemplo, isso exibirá `user.name`:

```
return (  
  <h1>  
    {user.name}  
  </h1>  
);
```

Você também pode “escapar para JavaScript” a partir de atributos JSX, mas é necessário usar chaves *em vez de* aspas. Por exemplo, `className="avatar"` passa a `"avatar"` string como classe CSS, mas `src={user.imageUrl}` lê o `user.imageUrl` valor da variável JavaScript e depois passa esse valor como `src` atributo:

```
return (  
  <img  
    className="avatar"  
    src={user.imageUrl}  
  />
```

Respondendo a eventos

Você pode responder a eventos declarando funções de *manipulador* de eventos dentro de seus componentes:

```
function MyButton() {  
  function handleClick() {  
    alert('You clicked me!');  
  }  
  
  return (  
    <button onClick={handleClick}>  
      Click me  
    </button>  
  );  
}
```

Observe como `onClick={handleClick}` não tem parênteses no final! Não *chame* a função do manipulador de eventos: você só precisa *transmiti-la*. O React chamará seu manipulador de eventos quando o usuário clicar no botão.

8.6 Eventos JavaScript

Um evento é a ocorrência de uma ação, geralmente produzida por um usuário, em uma página. Clicar em um botão, selecionar um item, sair de um campo, pressionar uma tecla, passar o mouse sobre uma imagem, redimensionar a página são alguns dos eventos que podem ser controlados em um sistema. Adicionar programação nas páginas web associadas à ocorrência dos diversos eventos JavaScript permite criar maior interatividade com o usuário, dando maior dinamismo à página. A partir da programação dos eventos, é possível, por exemplo, trocar uma imagem quando o usuário modifica a seleção de um item em uma lista de botões (Exemplo 9.1), exibir mensagens de advertência quando o usuário sai de um campo de edição com um conteúdo inválido ou, então, executar uma ação vinculada ao pressionamento de uma determinada tecla, além de vários outros.

A lista de eventos JavaScript passíveis de programação é grande. Eles podem estar relacionados com eventos de interface do usuário (`load`, `unload`, `resize`), eventos de mouse (`click`, `dblclick`, `mouseover`), eventos de teclado (`keypress`, `keydown`, `keyup`) ou eventos de formulário (`change`, `focus`, `blur`). A sintaxe para definir a programação de um evento é semelhante à utilizada a partir do segundo capítulo deste livro, modificando apenas o nome do evento que ficará associado a uma função ou que define a execução de uma função anônima.

Atualizando a tela

Freqüentemente, você desejará que seu componente “lembre” algumas informações e as exiba. Por exemplo, talvez você queira contar o número de vezes que um botão é clicado. Para fazer isso, adicione *estado* ao seu componente.

Primeiro, importe `useState` do React:

```
import { useState } from 'react';
```

Agora você pode declarar uma *variável de estado* dentro do seu componente:

```
function MyButton() {  
  const [count, setCount] = useState(0);
```

Você obterá duas coisas `useState`: o estado atual (`count`) e a função que permite atualizá-lo (`setCount`). Você pode dar a eles qualquer nome, mas a convenção é chamá-los como `[something, setSomething]`.

A primeira vez que o botão for exibido, `count` será `0` porque você passou `0` para `useState()`. Quando você quiser mudar de estado, chame `setCount()` e passe o novo valor para ele. Clicar neste botão incrementará o contador:

```
function MyButton() {  
  const [count, setCount] = useState(0);  
  
  function handleClick() {  
    setCount(count + 1);  
  }  
  
  return (  
    <button onClick={handleClick}>  
      Clicked {count} times  
    </button>  
  );  
}
```

O React chamará sua função de componente novamente. Desta vez, `count` será `1`. Então será `2`. E assim por diante.



useState (estado de uso)

hook



```
const [estado, setEstado] = useState(estado inicial)
```



*Variável a ser
utilizada no
componente*



*Função para
definir / alterar o
valor da variável*



*Valor inicial
da variável
de estado*



CAPÍTULO 8

Funções e eventos

Neste capítulo, vamos avançar no processo de criação de funções em um programa e no uso dos eventos JavaScript. Lidamos com as funções e os eventos desde o Capítulo 2, porém na sua forma básica. Todas as programações foram associadas, no geral, ao evento submit do form a partir da declaração de uma arrow function (função de seta). É possível criar funções com passagem de parâmetros, criar funções anônimas, fazer uma função devolver um determinado valor ou um conjunto de valores e, ainda, utilizar uma função como um módulo que contém um trecho de código que se repete e que pode ser acionado em vários pontos do programa. Assim, cada vez que precisamos executar esse trecho de código, chamamos o módulo, sem a necessidade de reescrever seus comandos.

Os eventos também podem ser mais bem explorados, pois há vários outros além do submit e click trabalhados até aqui nos exemplos do livro, como change, blur, focus, keypress... Eles ocorrem e podem ser programados quando, por exemplo, o usuário trocar um campo de seleção, sair de um campo de edição, posicionar o cursor em um elemento do formulário, ou, então, pressionar uma determinada tecla no preenchimento de um campo. Vamos verificar como podemos controlar melhor a execução de um programa ao adicionar ações associadas a esses eventos nos exemplos deste capítulo.

Sobre o processo de modularização de programas, é importante destacar um aspecto que vai facilitar a compreensão desse recurso: os módulos servem para melhor organizar nosso código, permitir a reutilização de trechos de programa e facilitar a sua leitura e manutenção.

8.1 Functions e Arrow Functions

A partir do momento em que nossos programas vão crescendo em tamanho e complexidade, dividi-los em blocos de código menores, ou seja, em funções, apresenta inúmeras vantagens. Pode-se destacar a possibilidade de reaproveitamento de código, a melhor organização e maior facilidade em entender um problema grande dividindo-o em blocos menores (dividir para conquistar), além dos benefícios relacionados ao trabalho em equipe.

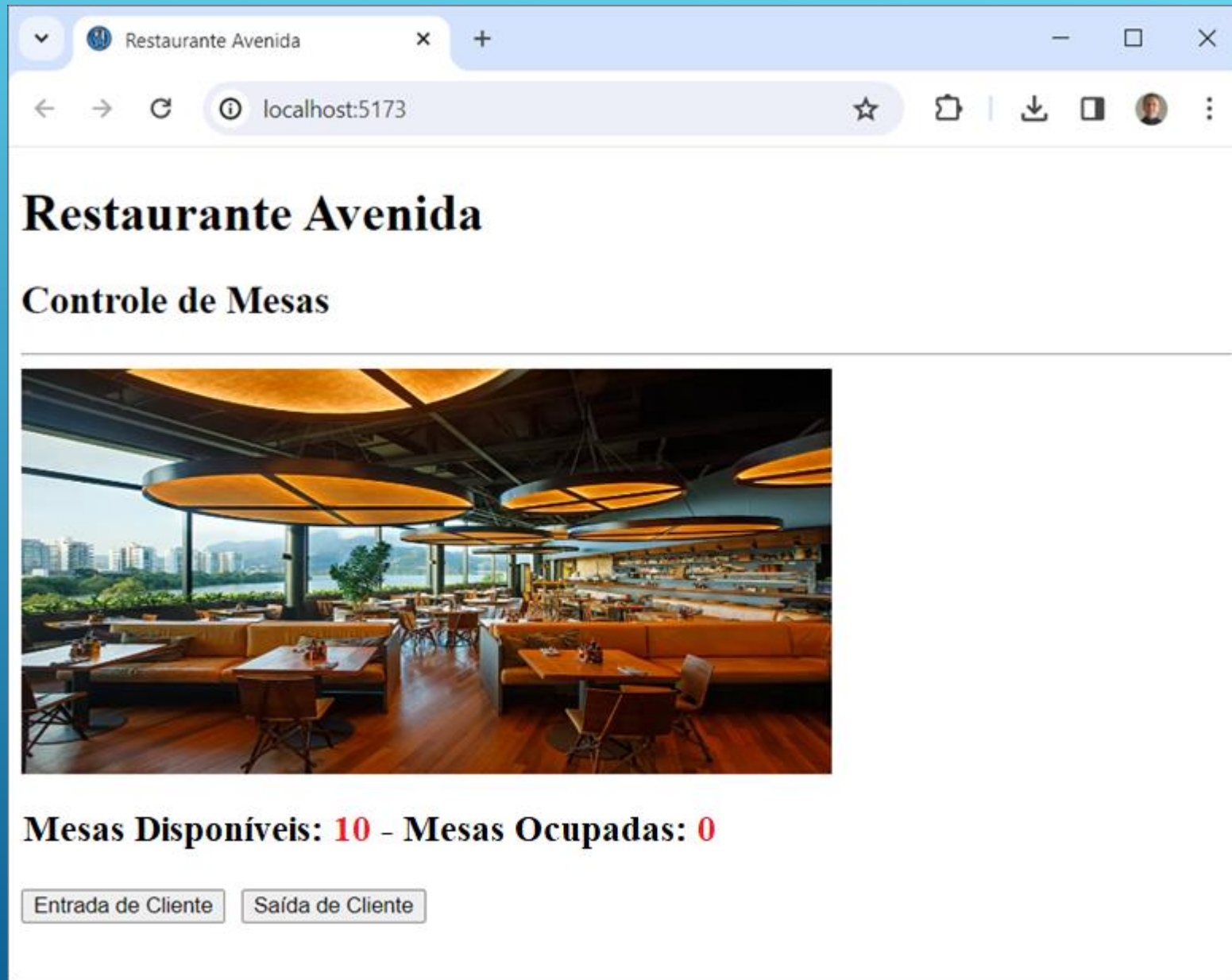
As próprias linguagens de programação são organizadas sobre essa lógica: há funções (métodos) para, por exemplo, verificar se um determinado conteúdo consta em um vetor, converter os caracteres de uma string em maiúsculas e classificar em ordem crescente uma lista de nomes. Imagine se você, toda vez que necessitasse ordenar uma lista de produtos a serem exibidos em um programa, tivesse de fazer isso comparando item por item? Melhor chamar o método que realiza esse processo e que já foi testado inúmeras vezes.

Uma função em JavaScript pode ser construída com a palavra reservada `function` ou como uma declaração de constante – usando uma notação conhecida como arrow function (função de seta), em que a função é atribuída para uma variável. Observe a sintaxe das duas formas:

```
<script>
  function ola() {
    alert("Olá. Seja muito bem-vindo!")
  }
  ola()

  const ola2 = () => {
    alert("Olá. Seja muito bem-vindo, novamente!")
  }
  ola2()
</script>
```


Exemplo: Página interativa e conceitos do React



Exercício:

Criar um novo projeto em React que exiba o layout conforme indicado inicialmente abaixo. Exibir mensagem conforme o clique do usuário em Casa ou Apartamento.

